Non-intrusive *sparse subspace learning* for parametrized problems

Domenico Borzacchiello · José V. Aguado · Francisco Chinesta

Abstract We discuss the use of hierarchical collocation to approximate the numerical solution of parametric models. With respect to traditional projection-based reduced order modeling, the use of a collocation enables non-intrusive approach based on sparse adaptive sampling of the parametric space. This allows to recover the low-dimensional structure of the parametric solution subspace while also *learning* the functional dependency from the parameters in explicit form. A sparse low-rank approximate tensor representation of the parametric solution can be built through an incremental strategy that only needs to have access to the output of a deterministic solver. Non-intrusiveness makes this approach straightforwardly applicable to challenging problems characterized by nonlinearity or non affine weak forms. As we show in the various examples presented in the paper, the method can be interfaced with no particular effort to existing third party simulation software making the proposed approach particularly appealing and adapted to practical engineering problems of industrial interest.

Keywords Reduced order modeling · Non-intrusiveness · Low rank approximations · Sparse identification · Sparse subspace learning · Hierarchical collocation

This work has been supported by ESI GROUP through the ECN-ESI Chair on advanced modeling and simulation of materials, structures and processes.

D. Borzacchiello, J.V. Aguado & F. Chinesta
High Performance Computing Institute &
ESI GROUP Chair @ Ecole Centrale de Nantes
1 rue de la Noe, F-44300 Nantes, France
E-mail: {domenico.borzacchiello;jose.aguado-lopez;francisco.chinesta}@ec-nantes.fr
*Corresponding author: D. Borzacchiello

1 Introduction

1.1 Motivation

Models are often defined in terms of one or several parameters regardless of the nature of the phenomenon they describe. Parameters allow to adaptively adjust a general model to a wide range of different scenarios governed by the same physical laws. In continuum mechanics, for instance, models are expressed mathematically as partial differential equations (PDE) representing the balance or the conservation laws governing the evolution of the fundamental variables that describe the state of a system. Numerical methods for the solution of such differential models have reached in many cases a remarkable degree of maturity, enabling the development of a variety of simulation software. This technological advancement has resulted in a massive usage of the simulation tools at the industrial scale, mainly as a means of solving practical problems such as design, optimization or inverse identification [**?**]. In this context, the user is usually interested in assessing the adequacy of a particular configuration with regard to a given design or performance criterion. This means in practice that the user decides, based on simulation results, whether a particular design fulfills or not the product specifications, or, in an optimization context, whether the process performance is improved or worsened by a specific setup. The decision-making process based on simulation commonly referred to as *simulation-based engineering* [**?**,**?**].

In spite of the simplicity of this concept, simulation-based engineering may be somehow limited by the need of evaluating multiple configurations before converging to a satisfactory result, that is, a design that fulfills the product specifications or a setup that optimizes the process performance. This practice, in which simulations have to be carried out for several changing configurations is the foundation of *multi-query* simulation. Each query (i.e. configuration evaluation) requires allocating some time to prepare a new simulation, execute it and analyze the results. Delays due to communication and information exchange in collaborative work projects should also be taken into account. This may render the design, or optimization, process very time consuming, thus only allowing for the evaluation of few configurations. Even though simulation-based engineering can be to some extent automated, the amount of configurations that can be evaluated within *reasonable* time remains very low when compared to the amount of potential configurations, which is potentially disproportionate [**?**].

In this paper, we shall assume that the ensemble of potential configurations of the system under study can be explicitly defined in terms of a set of parameters varying in a given interval. These parameters may be related to the material's behavior, the geometry or boundary conditions definition, for instance. In addition, parameters are assumed to be uncorrelated, which provides the parametric domain with a cartesian structure. We shall note by $\boldsymbol{\mu} \in \mathcal{M} \subset \mathbb{R}^D$ a D-tuple parameter array in the parametric domain $\mathcal{M}$. Since real applications are usually defined in terms of several parameters, parametrized problems can be seen in fact as high-dimensional problems, due to the need of large-scale exploration a high-dimensional parametric domain. This explains why brute-force approaches based on extensive "grid search" are generally precluded.

1.2 Computational methods for parametrized problems

Many methods have been developed in order to address the problem of dimensionality by proposing strategies for parsimonious exploration of the parametric domain. Design of Experiments (DoE), widely used in the industry, is a very practical approach that can help reducing the number of simulations based on a series of statistical indicators [**?**]. In the context of optimization, the parametric domain exploration may be guided by some optimization algorithm, such as gradient-based methods or any of their numerous variants [**?**]. The optimization algorithm has to be chosen carefully, according to the properties and structure of the cost function at hand.

An alternative approach for addressing parametrized problems is based on reduced order modeling methods (ROM). Since most of the time is consumed in multi-query simulation, ROM methods are designed so as to reduce the computational complexity of evaluating a given configuration. This is usually achieved by splitting the computational cost by means of an "offline-online" strategy. The key idea behind ROM is to build a problem-specific, low-dimensional subspace, that may either be learnt from available simulation data or computed by setting up an optimization problem. In the next subsections we briefly review both approaches as well as the bottlenecks in their practical implementation.

1.2.1 A posteriori reduced order modeling

Approaches based on learning from available simulation results are also known as *a posteriori* ROM methods. They typically require inspection of the parametric domain in the offline stage. This consists in carrying out simulations for different parameter samples using a full-order solver (based on finite elements for instance). Data collected in the learning stage is used to extract a reduced basis which is assumed to span the entire solution subspace for any parameter choice [**?**,**?**,**?**]:

$$\forall \boldsymbol{\mu} \in \mathcal{M}: \quad u(\boldsymbol{x};\boldsymbol{\mu}) \approx u^N(\boldsymbol{x};\boldsymbol{\mu}) \in V_N := \text{span}\{\phi^i(\boldsymbol{x})\}_{i=1}^N, \tag{1}$$

where $u(\boldsymbol{x};\boldsymbol{\mu})$ and $u^N(\boldsymbol{x};\boldsymbol{\mu})$ stand for the solution of some parametrized model and its approximation onto the low-dimensional subspace, namely V_N. In the adopted notation, the semicolon symbol is used to differentiate physical coordinates, $\boldsymbol{x}$, from parameters $\boldsymbol{\mu}$.

Note that in practice V_N is contained in W_M, an approximation space of dimension M used by the full-order solver. Provided that $N \ll M$, a reduced-order model can be built by projecting the residual of eq. (1) in the sense of Galerkin, onto the reduced subspace V_N. In this way the parametric solution is implicitly expressed by the set of reduced equations in terms of the coordinates of the solution onto the low-dimensional subspace.

Therefore in the online stage, if the solution is required for a new parameter configuration, a small algebraic system needs to be solved. Although this strategy results in a remarkable computational time save, a number of questions remain open:

– **Sampling**. Offline sampling of the parametric space is of critical importance. On one hand, sampling has to be fine enough to ensure that the structure of the

underlying subspace is captured within the level of desired accuracy. On the other hand, extensive sampling is usually unachievable in high-dimensional problems. In the context of the reduced basis method, an error estimator is used so as to guide the sampling process [**?**]. Error estimation evaluation in nonlinear problems is still object of ongoing studies [**?**].

- **Affinity**. Parameters must define an affine transformation. If that is not the case, the advantages of the offline-online strategy are compromised. Interpolation techniques have been proposed so as to overcome this issue [**?**], although a careful treatment is needed in order to preserve the spectral properties of the full-order operators [**?**].
- **Nonlinearity**. Nonlinear problems are a particular example of loss of affinity. Indeed, the evaluation of the non-linear terms entails a computational complexity which is in the order of that of the original non-reduced model [**?**]. Several approaches exist including empirical interpolation techniques [**?**,**?**], hyper-reduction [**?**,**?**], empirical quadrature rules [**?**] and collocation approaches [**?**].
- **Intrusiveness**. ROM methods implementation in third-party simulation software is quite intrusive since it requires access to the discrete operators arising from the discretization of the partial differential equations, which is not always possible [**?**].

1.2.2 A priori reduced order modeling

A priori ROM methods propose an alternative approach for parametrized problems. They are designed so as to compute the solution of the parametrized problem as an explicit function of the parameters. Therefore, a priori ROM methods do not produce a reduced system of equations but the *parametric solution* itself. Note that in this regard, parameters are treated exactly the same as if they were coordinates. Hence, we shall drop the semicolon indicating the parametric dependence and note: $u(\boldsymbol{x}, \boldsymbol{\mu})$. As a consequence, the computational domain becomes of higher dimension, as it must cover not only the physical and/or time coordinates, but also the parametric domain [**?**]. Since grid-based discretization is precluded in higher dimensional problems, tensor subspaces must be introduced to represent parametric solutions efficiently [**?**,**?**,**?**]. Several tensor subspaces have been proposed, see [**?**] for details. For the sake of brevity, we shall only present here the canonical tensor subspace, which lies at the basis of the Proper Generalized Decomposition (PGD) method [**?**,**?**,**?**]. Hence, the parametric solution is sought as:

$$u(\boldsymbol{x}, \boldsymbol{\mu}) \approx u^N(\boldsymbol{x}, \boldsymbol{\mu}) \in T_N := \text{span}\{\, \phi_0^i(\boldsymbol{x})\, \phi_1^i(\mu_1) \cdots \phi_D^i(\mu_D)\, \}_{i=1}^N, \tag{2}$$

where T_N is the subspace of tensors with canonical rank equal to N. Note that in practice $T_N \subset W_{M_0} \otimes W_{M_1} \otimes \cdots \otimes W_{M_D}$, where W_{M_d} is a generic approximation space (finite element like) of dimension M_d, with $0 \leq d \leq D$. In addition, ϕ_d^i are separated functions of each coordinate, usually called *modes*. Recall that this particular structure is possible thanks to the cartesian structure that has been assumed for the parametric domain. Note that the complexity of the tensor subspace defined in Eq. (2) grows linearly with the number of dimensions, while the complexity of a grid-based discretization grows exponentially.

While a posteriori ROM methods extract a subspace from data, a priori methods are designed so as to build a tensor subspace by setting up an optimization problem, that is, no sampling of the parametric domain is in principle required. In particular, PGD allows building a tensor subspace such as the one shown in Eq. (2) progressively by computing rank-one corrections, i.e. by building a series of nested subspaces:

$$T_1 \subset T_2 \subset \cdots \subset T_N \quad \text{where} \quad T_N := T_{N-1} + T_1. \tag{3}$$

In order to compute each rank-one correction, the weak form of the PDE at hand is regarded as an optimization problem where the set of admissible solutions is constrained to T_1. This yields a nonlinear optimization problem due to the tensor multiplicative structure of the subspace, see Eq. (2), which can be efficiently solved by applying an alternating directions algorithm [**?**]. In this algorithm, the optimization is performed alternatively along each dimension until convergence [**?**,**?**]. In this way, the algorithm splits a high-dimensional problem into a series of low-dimensional ones, achieving linear complexity in the dimensionality of the problem.

A priori ROM allows fast online exploration of several configurations, since evaluating a single parameter choice demands no more than reading a look-up table. Therefore, parametric solutions can be used as a black-box simulation database, to be easily integrated as a part of complex systems such as real time simulators [**?**] or Simulation Apps [**?**]. Other applications in which parametric solutions fit perfectly are fast process optimization [**?**] and real-time inverse identification and control [**?**].

Aside from the question of the parametric domain sampling, which is completely encompassed by a priori methods, both a priori and a posteriori approaches share essentially the same unresolved questions and challenges already discussed above: intrusiveness, affinity and by extension nonlinearity. The only differences rely on the technicality intrinsic to a tensor framework. The requirement for affinity translates in the need of a *tensor structure* in the problem setting. Without this, the alternated optimization of the modes cannot be performed in linear complexity. The same issue is encountered with non-linear problems: the tensor structure of the non-linear terms is in general not known a priori, and therefore, their evaluation becomes extremely expensive (comparable to that of a grid-based discretization). This issues make the integration of a priori ROM methods in third-party simulation software seems impractical unless major modifications in the original code are implemented.

1.3 Contributions and paper structure

From the discussion in the previous sections the following conclusion can be drawn: the main bottleneck of projection-based reduced order modeling can be resumed in that not only the solution must be *reducible* but also the problem formulation must have a proper structure.

In an attempt to encompass this limitation, we propose a *sparse subspace learning* (SSL) method for computing parametric solutions, based on the following main ingredients:

- Collocation approach in the parametric domain to circumvent the need of affinity or tensor structure in the problem setting, since the method is no longer based on Galerkin projection.

– Hierarchical basis providing nested collocation points (i.e. greedy sampling) and built-in error estimation capabilities.
– Incremental subspace learning to extract a low-rank approximation throughout the greedy collocation refinement process.
– Sparse adaptive sampling to extend hierarchical collocation to higher dimensions.

The SSL method is able to produce parametric solutions based only on the output of a deterministic solver to which the parameters are fed as an input. SSL can be coupled, in principle, to any third-party simulation software. The affinity of the parameter transformations is no longer an issue, as well as the treatment of nonlinearity which is entirely handled by the direct solver. Parametric sampling is intrinsically parallel. Although this feature is not explored in this paper, it makes this approach compatible with parallel and high performance computing.

The rest of the paper is organized as follows. In section 2 we illustrate the implementation of SSL and a detailed application example. In section 3 we extend the discussion to transient and pseudo-transient problems. Multi-parametric problems are treated in section 4, where a practical application to a three dimensional transient car crash model with four parameters is also presented. In the given examples we use both academic and commercial software to point out the ease of integrating the proposed approach using existing simulation software in a black-box fashion. Technical details on the incremental subspace learning are available in appendix A.

2 Sparse Subspace Learning

In this section we introduce the main contribution of this paper, the *sparse subspace learning* (SSL) method.

2.1 Underlying idea

Consider the solution of a parametrized problem is the scalar field $u(\mathbf{x}, \boldsymbol{\mu})$, with $\mathbf{x} \in \Omega$ and $\boldsymbol{\mu} \in \mathcal{M}$, the physical and parametric domains respectively. Given the approximation spaces

$$W_{N_s} := \text{span}\{\phi^i(\boldsymbol{x})\}_{i=1}^{N_s} \quad \text{and} \quad Z_{N_\mu} := \text{span}\{\psi^j(\boldsymbol{\mu})\}_{j=1}^{N_\mu},$$

its numerical approximation can be expressed as:

$$u(\mathbf{x}, \boldsymbol{\mu}) \approx u^h(\mathbf{x}, \boldsymbol{\mu}) := \sum_{i=1}^{N_s} \sum_{j=1}^{N_\mu} S_{ij} \phi^i(\mathbf{x}) \psi^j(\boldsymbol{\mu}) \,.$$

The scalars S_{ij} are *representation coefficients* computed trough numerical method after appropriate discretization of the equations governing the problem in consideration. The complexity of this representation is $N_s \times N_\mu$, with N_s and N_μ being the number of degrees of freedom in the physical and parametric space.

In the framework of parametric reduced order modeling, the notion of *reducibility* of the solution has a two-fold meaning [**?**,**?**]. In particular, it requires that the matrix of representation coefficients $\mathbf{S}$ have two structural properties:

– **Sparsity**. This roughly translates into asking that most of the coefficients in $\mathbf{S}$ vanish, or in practice that there are a few dominant components whose magnitude is much greater than the rest. The sparsity, ζ, is defined as the number of vanishing coefficients of $\mathbf{S}$ per row. Hopefully, $\zeta \sim N_\mu$.
– **Low-rank**. The parametric solution lives in a lower dimensional subspace of W_{N_s}, that is, $r \ll \min(N_s, N_\mu)$. This basically consists in the assumption that only few columns of $\mathbf{S}$ are linearly independent.

Through a collocation strategy it is possible to determine the low rank structure, by extracting a reduced basis from the solutions at sampled points, while simultaneously capturing the functional dependency from the parameters, in order to have numerical parametric solution in explicit form. Parametric collocation approach seeks to compute representation coefficients S_{ij} as a linear combination of sampling or *measurements* of the sought function evaluated at particular points $\{\mathbf{x}_i\}$ and $\{\boldsymbol{\mu}_l\}$:

$$M_{il} := u^h(\mathbf{x}_i, \boldsymbol{\mu}_l)\,.$$

Data can be structured into a matrix $\mathbf{M}$ which is a collection of output of a deterministic solver (sometimes called the snapshots matrix), each column containing the solution computed for a specific value of input parameters $\boldsymbol{\mu}$. In the following we assume that the solver output is a reliable approximation of the true solution within a controllable tolerance, that is, the discretization of the physical space N_s is sufficient to guarantee the desired accuracy.

In practice, the relation between the representation and measurements coefficients is given by a linear application that writes as:

$$\mathbf{SR} = \mathbf{M}\,,$$

where the matrix $\mathbf{R}$ is a linear operator mapping the rows of $\mathbf{S}$ into the rows of $\mathbf{M}$. Once the snapshots matrix is available, the representation coefficients are obtained by applying $\mathbf{R}^{-1}$ to the rows of $\mathbf{M}$. This operation is potentially costly since $\mathbf{R}$ is a large dense matrix that needs updating each time the parametric space is refined (i.e. new columns are appended to the snapshots matrix). The transformation matrix results from the choice of the representation basis and the sampling points, since

$$R_{jl} = \psi^j(\boldsymbol{\mu}_l)\,.$$

Therefore, an optimal choice of the sampling strategy to select the points $\{\boldsymbol{\mu}_l\}$ and the representation basis $\psi^j(\boldsymbol{\mu})$ would lead ideally to the highest possible sparsity s while giving rise to an operator $\mathbf{R}$ that is easily invertible and *updatable* as the sampling is refined. This concept is exploited in the next section using hierarchical parametric collocation.

2.2 Discovering sparsity through hierarchical collocation

The most intuitive way to apply collocation in the parametric space is by using an interpolative basis. This choice results into

$$\mathbf{S} \equiv \mathbf{M}$$

because the linear operator **R** is the identity matrix in this case. Although the number of sampling can be adjusted to control the convergence in the parametric sampling $\{\boldsymbol{\mu}_l\}$, interpolative basis do not in general yield sparsity of the solution. On the other hand, spectral basis, such as polynomials or trigonometric functions, guarantee sparsity, provided that the solution have the required regularity to be represented in the chosen basis. When that is the case, spectral coefficients decrease exponentially with the order of approximation [**?**]. The price to pay for sparsity is that the operator **R** is full, because spectral basis are not interpolative. This implies that the computational cost needed to compute the representation coefficients becomes impractical for large systems and high dimensional parametric problems, for which the number of required sampling is high and each sampling evaluation is costly. Furthermore, if refinement is needed, **R** must be updated and a new linear system has to be solved again.

To circumvent this problem there are two possible approaches:

- **Subsampling**. By choosing the number of representation functions $\psi^j(\boldsymbol{\mu})$ to be higher than the number of sampled solutions (**S** has more columns than **M**) one obtains an over-determined system. Through an appropriate regularization and choice of the sampling points, which must be incoherent with respect of representation basis [**?**], a unique solution can be determined. This is the approach followed by Sparsity Promoting techniques [**?**] Compressed Sampling [**?**] and Basis Pursuit [**?**]. In practice, regularization is ensured by asking the maximum sparsity in the solution, which is equivalent in minimizing the zero-norm of each column in **S**. This problem is NP-hard and therefore a surrogate norm is chosen instead of the norm zero, often the norm-1, leading to a smooth convex functional. The minimization step can be performed using many of the techniques specifically developed for this problem [**?**,**?**]. Once all the coefficients are computed the ones with the lowest magnitude are simply purged. Although this approach is extremely efficient in terms of the number of sampling measurements needed, the computation of these coefficients becomes exponentially harder as the number of parameters increases [**?**].
- **Hierarchical Collocation**. In this approach the representation coefficients are not computed all at once but through and incremental procedure based on multilevel sampling. This idea is used in Sparse Grid [**?**] and Stochastic Collocation [**?**] using a hierarchical sampling based on a sequence of nested sets of points. In this way the operator **R** is quasi-interpolative because it is block-triangular. This means that the computation of the sparse coefficients in **S** does not require the solution of an algebraic system, or the minimization of a nonlinear functional, but can be computed relatively easily directly from the solution samplings.

Hierarchical sampling is the approach followed in this paper. This is based on the definition of a hierarchy of collocation points sets. At level k of the sampling hierarchy, the corresponding set of points has $N_\mu^{(k)}$ elements.

$$\mathcal{P}^{(k)} \equiv \{\boldsymbol{\mu}_l\}_{l=1}^{N_\mu^{(k)}}$$

In hierarchical collocation the point sets are nested:

$$\cdots \subset \mathcal{P}^{(k-1)} \subset \mathcal{P}^{(k)} \subset \mathcal{P}^{(k+1)} \subset \cdots \quad \forall k \in \mathbb{N},$$

This implies that each level contains the $N_\mu^{(k-1)}$ points of the previous level plus $N_\mu^{(k)} - N_\mu^{(k-1)}$ additional points. The subsets of new points that are progressively added are called *hierarchical refinements*:

$$\mathcal{H}^{(k)} \equiv \mathcal{P}^{(k)} \setminus \mathcal{P}^{(k-1)} \quad \forall k \in \mathbb{N}^+ .$$

The nested structure of sampling sets also entails that the subspaces

$$Z^{(k)} := \text{span}\{\psi^j(\boldsymbol{\mu})\}_{j=1}^{N_\mu^{(k)}}$$

are also nested:

$$\cdots \subset Z^{(k-1)} \subset Z^{(k)} \subset Z^{(k+1)} \subset \cdots \quad \forall k \in \mathbb{N} .$$

In hierarchical collocation the basis functions are quasi-interpolative in the sense that, for a given a hierarchical level k and $N_\mu^{(k-1)} < j \leq N_\mu^{(k)}$

$$\psi^j(\boldsymbol{\mu}_l) \begin{cases} = 0 & \text{if } \boldsymbol{\mu}_l \in \mathcal{P}^{(k-1)} \\ = \delta_{jl} & \text{if } \boldsymbol{\mu}_l \in \mathcal{H}^{(k)} \\ \neq 0 & \text{otherwise} \end{cases} \tag{4}$$

This is essential in order to ensure the block triangular structure of the matrix $\mathbf{R}$, which is the key feature in order to have an easy direct solution of the linear system for the determination of the representation coefficients. Indeed, in reason of the particular structure of $\mathbf{R}$, back substitution can be used to compute the columns of $\mathbf{S}$:

- For the first level $k = 0$, $0 < j \leq N_\mu^{(0)}$, the representation coefficient are simply equivalent to the snapshots : $M_{ij} = S_{ij}, \forall i = 1, \ldots, N_s$
- For subsequent levels $k > 0$, $N_\mu^{(k-1)} < j \leq N_\mu^{(k)}$, the following explicit formula is used

$$S_{ij} = M_{ij} - \sum_{l=1}^{N_\mu^{(k-1)}} S_{il}\, \psi^l(\boldsymbol{\mu}_j) \quad \forall i = 1, \ldots, N_s \tag{5}$$

The previous equation is applied recursively over the hierarchical levels using few simple algebraic operations and without the need of updating the previously computed S_{ij} coefficients. Indeed, the second term in the right hand side of (5) corresponds to the predicted values from interpolation of the previously computed snapshots $u^h(\boldsymbol{x}, \boldsymbol{\mu}_j)$ onto the sampling points of the new hierarchical refinement

$$\bar{M}_{ij} = \sum_{l=1}^{N_\mu^{(k-1)}} S_{il}\, \psi^l(\boldsymbol{\mu}_j) \quad \forall i = 1, \ldots, N_s\, , \boldsymbol{\mu}_j \in \mathcal{H}^{(k)}\, ;\, k \in \mathbb{N}^+ \tag{6}$$

Therefore, we conclude that the representation coefficients S_{ij} are simply recovered as the difference between the actual function values M_{ij} and the *predictions* $\bar{M}_{ij}$:

$$S_{ij} = M_{ij} - \bar{M}_{ij} .$$

For this reason, in the context of hierarchical collocation, the representation coefficients S_{ij} are also called the *surplus* coefficients. The corresponding functions

$$s(\mathbf{x}; \boldsymbol{\mu}_j) = \sum_{i=1}^{N_s} S_{ij} \phi^i(\mathbf{x})$$

are consequently named the *surplus functions*.

The matrix $\mathbf{R}$ is never formed or inverted in practice, which is why the method is easily implemented in a greedy incremental way. The termination criterion is simply based on the convergence of the surplus functions in a given norm, like for instance the euclidean norm

$$\left\| s(\mathbf{x};\boldsymbol{\mu}_j) \right\|_2 = \sqrt{\sum_{i=1}^{N_s} S_{ij}^2}\,.$$

Alternatively the ℓ^2 − norm or the infinity norm can be also used. The algorithm is stopped when the norm of all the surplus functions in a hierarchical level falls below a given tolerance ϵ_h. This constructive approximation is described in algorithm 1.

Algorithm 1 Hierarchical sampling: `HS`

1: Set $S_{ij} \equiv M_{ij}$ for $k = 0$
2: **for** $k = 1$ to $L_{\max}$ **do**
3: Compute predictions $\bar{M}_{ij}$ using Eq. (6)
4: Run $M_{ij} = u^h(\boldsymbol{x}_i, \boldsymbol{\mu}_j)$ ▷ "True" solution
5: Compute $S_{ij} = M_{ij} - \bar{M}_{ij}$ ▷ Surplus coefficients
6: Check $\left\| s(\mathbf{x};\boldsymbol{\mu}_j) \right\| < \epsilon_h$ for all $\boldsymbol{\mu}_j$ in current level
7: **end for**

To discover the sparsity in the parametric space, algorithm 1 requires the solution at all sampling points, even though only few relevant surplus coefficients are retained in the end. Reduction is obtained "a posteriori", by pruning negligible coefficients once the snapshots M_{ij} are already computed. This means that most of snapshots are not actually used because they lead to small coefficients S_{ij} which are inevitably pruned. In order to avoid computing useless direct solutions, it is possible to introduce the of concept adaptive sampling.

In hierarchical multivariate interpolation and quadrature, this concept has been extensively investigated and rigorous error estimation strategies exist in order to adapt the sampling procedure while retaining a given accuracy [**?**,**?**,**?**]. In the present study the functions that we are trying to approximate are the solutions of parametric partial differential equations. Therefore residual can be computed explicitly at new sampling points from the prediction $\bar{M}_{ij}$ given by the interpolation of previous hierarchical levels and used to define a sampling strategy based on a proper error estimator. In practice the residual based error estimation checks whether the predicted values are accurate enough. In case of positive outcome a new direct solution is not required and the corresponding sampling point can be simply skipped. Based on this basic sampling adaptivity strategy, the adaptive algorithm 2 can be constructed.

The benefits of the adaptive strategy will be further discussed in the examples presented in section 2.4.3.

Algorithm 2 Adaptive Hierarchical Sampling: `AHS`

Set $S_{ij} \equiv M_{ij}$ for $k = 0$
for $k = 1$ to $L_{\max}$ **do**
  Compute predictions $\bar{M}_{ij}$ using Eq. (6)
  Compute $E_{ij} = e(\bar{M}_{ij})$ ▷ Residual error estimation
  if $E_{ij} > \epsilon_{\text{tol}}$ **then**
    Run $M_{ij} = u^h(\boldsymbol{x}_i, \boldsymbol{\mu}_j)$ ▷ $\bar{M}_{ij}$ as first guess solution
    Compute $S_{ij} = M_{ij} - \bar{M}_{ij}$ ▷ Surplus coefficients
  else
    Set $S_{ij} = 0$
  end if
  Check $\left\| s(\mathbf{x}; \boldsymbol{\mu}_j) \right\| < \epsilon_h$ for all $\boldsymbol{\mu}_j$ in current level
end for

2.3 Incremental subspace learning

Although the hierarchical collocation approach provides many advantages, the final parametric solution is not optimal in terms of its compactness. As a matter of fact, the number surplus functions needed to approximate the parametric solutions is in principle equal to the number of hierarchical collocation points. In most applications, it is likely that a low rank approximation of the solution exists. The structure of the low-dimensional subspace for the hierarchical surpluses can be *learned* using approximate low-rank tensor decomposition techniques on the matrix $\mathbf{S}$. This is especially important when one is interested in solving large problems, which may lead to thousands of full-order solves, especially when multi-parametric problems are to be addressed. Indeed, the storage of the matrix $\mathbf{S}$ can be limited by memory availability when solving large-scale problems. In what follows, we shall denote a matrix $\mathbf{A}$ with values in the field $\mathbb{K}^{m \times n}$ either by $\mathbf{A}_{|m \times n}$, the first time it is introduced, or simply $\mathbf{A}$, in subsequent appearances. Therefore the representation coefficient matrix at hierarchical level k is $\mathbf{S}_{|N_s \times N_\mu^{(k)}}$. After k hierarchical refinements, we seek a matrix approximation of rank-r given by

$$\mathbf{S}_k \approx \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^*, \tag{7}$$

where $\mathbf{U}_{|N_s \times r}$ and $\mathbf{V}_{|N_\mu^{(k)} \times r}$ are orthonormal matrices and $\mathbf{\Sigma}_{|r \times r}$ is a diagonal matrix with nonnegative coefficients. Because of sparsity, the rank of $\mathbf{S}_k$ after k refinements is at most equal to the number of nonzero columns, that is, $N_\mu^{(k)} - \zeta$. The decomposition in equation (7) can be computed in practice by either truncating the singular value decomposition of $\mathbf{S}$ to the r highest singular values [**?**], or using randomized singular value decomposition (`rsvd`) [**?**], or constructive incremental approximations like `parafac` [**?**] and `candecomp` [**?**]. Low rank approximation can be performed as a post-processing step in "batch mode", after convergence of the sparse learning sampling. However, this approach is expensive in terms of both memory requirements and computational cost because it implies storing an manipulating $\mathbf{S}$ throughout the whole process of parametric refinement. On the other hand the size of $\mathbf{S}$ grows dynamically thought the adaptive hierarchical sampling procedure as additional columns are appended each time the parametric space is refined and new surplus functions are computed. Therefore the tensor decomposition technique should enable incremental

updating after each refinement on order computations from scratch. In this way hierarchical collocation can be coupled with a subspace extraction step that is performed on-the-fly as new surpluses are computed. The idea of *incremental subspace learning* is presented in several works and it is mostly used to compress continuously streaming data flows with application to computer vision and pattern recognition where on batch subspace extraction is unfeasible due to the sheer size of the datasets [**?**,**?**,**?**]. Once a matrix update $\mathbf{\Delta}_{k+1}$ is appended to the matrix $\mathbf{S}_k$ the following factorization is considered [**?**]:

$$\mathbf{S}_{k+1} = \left[\mathbf{S}_k \middle| \mathbf{\Delta}_{k+1}\right] \approx \left[\mathbf{U}_k \middle| \mathbf{J}_{k+1}\right] \left[\begin{array}{c|c} \mathbf{\Sigma}_k & \mathbf{L}_{k+1} \\ \hline \mathbf{0} & \mathbf{K}_{k+1} \end{array}\right] \left[\begin{array}{c|c} \mathbf{V}_k & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right]^* \tag{8}$$

where $\mathbf{L}_{k+1} = \mathbf{U}_k^* \mathbf{\Delta}_{k+1}$, $\mathbf{K}_{k+1} = \mathbf{J}_{k+1}^* \mathbf{H}_{k+1}$, $\mathbf{H}_{k+1} = (\mathbf{I} - \mathbf{U}_k \mathbf{U}_k^*)\mathbf{\Delta}_{k+1}$, $\mathbf{J}_{k+1}$ is an orthogonal basis for $\mathbf{H}_{k+1}$ and $\mathbf{I}$ is the identity matrix. The middle factor can be further decomposed as

$$\left[\begin{array}{c|c} \mathbf{\Sigma}_k & \mathbf{L}_{k+1} \\ \hline \mathbf{0} & \mathbf{K}_{k+1} \end{array}\right] \approx \tilde{\mathbf{U}}_{k+1} \tilde{\mathbf{\Sigma}}_{k+1} \tilde{\mathbf{V}}_{k+1}^*, \tag{9}$$

and finally the low rank decomposition is updated :

$$\mathbf{U}^{k+1} = \left[\mathbf{U}^k \middle| \mathbf{J}\right] \tilde{\mathbf{U}}^{k+1}, \tag{10}$$

$$\mathbf{V}^{k+1} = \left[\begin{array}{c|c} \mathbf{V}^k & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \tilde{\mathbf{V}}^{k+1}, \tag{11}$$

and

$$\mathbf{\Sigma}^{k+1} = \bar{\mathbf{\Sigma}}^{k+1}. \tag{12}$$

The low-rank tensor approximation in equation (9), can be obtained by means of any of the above mentioned techniques. The computational cost of the update only depends on the size of $\mathbf{\Delta}$ and not on the full dataset size. In this work we employ `rsvd` to perform this operation and to allow the rank to change throughout the refinement levels k. The resulting incremental version of the `rsvd` is therefore referred to as `irsvd`. For the sake of concision, implementation details are reported in appendix A.

The `irsvd` can be seen as an incremental subspace learning technique as it allows to update the low rank representation dynamically as new snapshots in the solutions are computed. It can be combined with hierarchical adaptive sampling into a *sparse subspace learning* (`ssl`) approach, in which the sampling of the parametric space simultaneously used to build reduced basis and discover the sparse structure of the subspace. The use of a low rank representation for $\mathbf{S}$ is not only beneficial in terms of memory usage but it also simplifies the hierarchical sampling algorithm. Indeed, when $\mathbf{S}$ is approximated using the canonical tensor decomposition (7), the interpolation of the surplus functions onto the sampling point of a new hierarchical level can be performed more efficiently. Instead of recombining the columns of $\mathbf{S}$ we interpolate the right (parametric) singular vectors $\mathbf{V}$:

$$\bar{V}_{ij} = \sum_{l=1}^{N_\mu^{(k-1)}} V_{il} \psi^l(\boldsymbol{\mu}_j) \tag{13}$$

Algorithm 3 Sparse Subspace Learning: SSL

For $k = 0$ $S_{ij} \equiv M_{ij}$.
Compute $\mathbf{S}_0 \approx \mathbf{U}_0\mathbf{\Sigma}_0\mathbf{V}_0{}^*$ ▷ rsvd
for $k = 1, 2, \ldots$ **do**
  Compute predictions $\bar{M}_{ij}$ using Eqs. (13),(14)
  Compute $E_{ij} = e(\bar{M}_{ij})$ ▷ Residual error estimation
  if $E_{ij} > \epsilon_{\text{tol}}$ **then**
    Run $M_{ij} = u^h(\boldsymbol{x}_i, \boldsymbol{\mu}_j)$ ▷ $\bar{M}_{ij}$ as first guess solution
    Compute $S_{ij} = M_{ij} - \bar{M}_{ij}$ ▷ Surplus coefficients
  else
    Set $S_{ij} = 0$
  end if
  if All $\left\|s(\mathbf{x};\boldsymbol{\mu}_j)\right\|_2 < \epsilon_h$ **then**
    break
  else
    Update $\mathbf{S}_k \approx \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k{}^*$ ▷ irsvd
  end if
end for

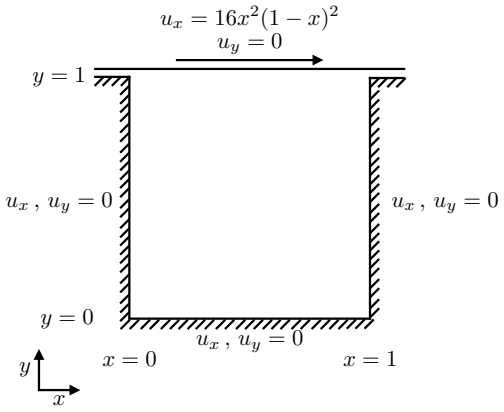


Fig. 1: Schematics for the 2D lid driven cavity flow.

and obtain the predictions at new sampling points as:

$$\bar{\mathbf{M}} = \mathbf{U}\mathbf{\Sigma}\bar{\mathbf{V}}^* \tag{14}$$

The cost of the interpolation step depends on the numerical rank r of the approximation which is much smaller than the number of surplus functions N_s in many practical applications. The resulting procedure is detailed in algorithm 3.

2.4 Parametric lid driven cavity flow

We consider a classic benchmark problem in computational fluid dynamics: the lid-driven cavity flow (fig 1). The choice of the benchmark is motivated by its mathe-

matical features that typically need a careful treatment in the framework of projection based parametric reduced order modeling, like nonlinearity and non-convexity [**?**]. Therefore it can be considered as a minimal working example to showcase the features of the sparse subspace learning approach. In particular we consider low Reynolds steady laminar flow governed by the equations :

$$\begin{cases} \mu\, \mathbf{u} \cdot \nabla \mathbf{u} - \Delta \mathbf{u} + \nabla p = \mathbf{0} \\ \nabla \cdot \mathbf{u} = 0 \end{cases}, \tag{15}$$

with $\mathbf{u}(\boldsymbol{x}, \mu)$ and $p(\boldsymbol{x}, \mu)$ being the velocity and pressure fields and $\mu = \frac{\rho V L}{\eta}$ being the Reynolds number. For this problem we seek a numerical approximation for $\mathbf{u}(\boldsymbol{x}, \mu)$, $(\boldsymbol{x}, \mu) \in [0, 1]^2 \times [0, 1000]$.

2.4.1 Deterministic solver

For a given choice of the Reynolds number the flow is solved with a high order mixed finite element formulation [**?**]. A cartesian mesh of 40 elements per direction is used. The nonlinearity is handled using a pseudo-transient method. Starting from an initial guess, that can be the corresponding Stokes flow for example, equation are integrated in time using a stable time-stepping algorithm until the steady state. Convergence is assessed using by checking the relative residual norm of eq. (15) and the ℓ_2 norm of the difference of the velocity field between two successive iterations.

2.4.2 Choice of the parametric basis

In low Reynolds number regime it can be reasonably assumed that the flow is laminar and that the solution of the equations (15) varies smoothly with the parameter μ. In this case we can adopt a high order polynomial basis to represent $\mathbf{u}(\boldsymbol{x}, \mu)$ in the parametric space for Reynolds numbers $\mathcal{M} \equiv [0, 1000]$.

When using polynomial approximation an optimal choice for the sampling is defined by the set of Gauss-Chebychev-Lobatto (GCL) points

$$\mathcal{P}^{(k)} \equiv \begin{cases} \{0, 1000\} & \text{if } k = 0 \\ \left\{\mu_j = 500(cos(\frac{2j-1}{2^k}) + 1) \forall\, j = 1, \ldots, 2^{k-1}\right\} \cup \mathcal{P}^{(0)} & \text{if } k > 0 \end{cases}$$

The corresponding parametric basis is constructed using hierarchical Lagrangian interpolation. For a given hierarchical level k and $N_\mu^{(k-1)} < j \leq N_\mu^{(k)}$:

$$\psi^j(\mu) \equiv \mathcal{L}_j^{(k)}(\mu) = \prod_{\mu_i \in \mathcal{P}^{(k)} \setminus \mu_j} \frac{\mu - \mu_i}{\mu_j - \mu_i} \quad \forall \mu_j \in \mathcal{H}^{(k)}$$

2.4.3 Results

In order to get better insight on the structure of the sparse representation of the solution in the parametric space, algorithm 1 is run first without adaptiveness in the sampling. Figure 2 shows a heatmap plot of the elements in the representation $\mathbf{S}$ and

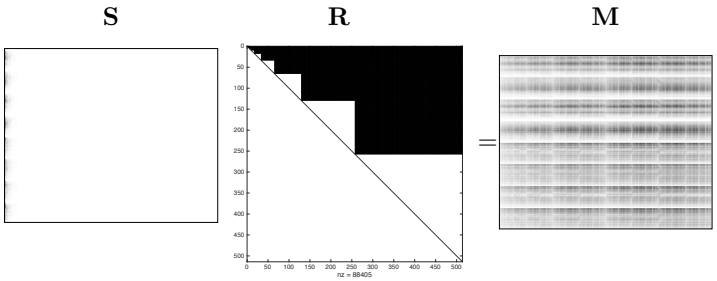


Fig. 2: Heatmap plot showing the magnitude of both sampling points M_{ij} and representation coefficients S_{ij}, evidencing the sparsity of the solution of the parametric lid-driven cavity flow in the space spanned by hierarchical polynomial function. The block triangular structure of the matrix **R** is also shown.

measurements **M** matrices to give a qualitative visualization of the sparsity of the of first with respect to the second. Indeed, more than half of the elements in **S** can be pruned without significant effect on the solution, that is with a relative approximation error less than 10^{-6}. The same figure shows the block diagonal structure of the interpolation operator **R** obtained from the quasi-interpolative feature of the hierarchical polynomial basis used in this example.

As a second step we solve the same problem with adaptive sampling. This can be introduced in a straightforward and non-intrusive way. Indeed, most of well designed iterative solvers (linear and nonlinear) perform a check on the initial guess solution before actually start running. If the convergence criterion is already met by the first attempt solution, the solver does not perform any iteration and returns the initial guess as the correct solution. Following this rationale, the predictions from hierarchical $\bar{M}_{ij}$ interpolation offers an optimal initial guess to initialize the deterministic solver at a new sampling point. If this solution is good enough the solver is not run and the corresponding surplus coefficient is automatically set to zero. This strategy avoids the delicate task of defining a specific error estimation approach for each new problem by leaving this issue to the deterministic solver which can be therefore easily plugged in the SSL algorithm. Running algorithm 2 confirms this idea. In figure 3 we report the magnitude of the hierarchical surpluses as a function of the simulation number (simulation are ordered in the sense of hierarchical levels). These decay very fast until they reach a plateau around the level of 10^{-8}. This phenomenon is explained by the fact that convergence criterion tolerance was set to this value, therefore the presence of the plateau indicates that the parametric sampling reached the level of precision of the direct solver. The points that are set to zero (to the machine precision) correspond to the cases where the adaptive algorithm skipped the solution and automatically set the corresponding surplus to zero. In this case, more than half of the solution were not computed at all.

As the parametric sampling is refined, the sparse surplus coefficients S_{ij} converge and the prediction $\bar{M}_{ij}$ becomes an increasingly better initial guess for M_{ij}. This implies that the convergence in the parametric space also accelerates the convergence

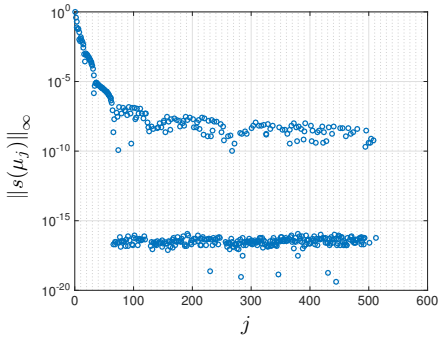


Fig. 3: The norm of the sparse surplus functions $s(\boldsymbol{x}, \mu_j)$ decreases with the refinement in the parametric space. The stagnation observed is due to the fact that the convergence tolerance in the direct solver is set to 10^{-8}, while the point with zero (to the machine precision) surplus coefficients correspond to the simulations that were skipped by the adaptive algorithm.

on the deterministic iterative solver which require less and less iterations to reach convergence. To quantify this idea, we represented a measure of the computational work associated to each direct solution performed by the solver as a function of the magnitude of the surplus coefficients, figure 4. We assume, as a measure of the computational work, the number of iterations required by the pseudo-transient solver to converge to the steady state. In this way we make this measure platform independent. In figure 4 the surplus coefficients are shown to be linearly correlated to the computational work in the asymptotic convergence regime. Since polynomial approximation guarantees exponential convergence of the S_{ij} coefficients, the linear decay of the computational work with the surplus coefficients implies that even though the number of sampling points increase the overall computational work per hierarchical level decreases.

The numerical rank obtained in the final approximation using a tolerance of 10^{-7} in the irsvd algorithm is $r = 9$. Both space functions and parameters functions are represented in figure 5. For further details see appendix A.

3 Time dependent models

Parametric solutions of transient, or pseudo-transient, models can also be computed in the SSL framework. A first possibility consists in considering a space-time discretization

$$u^h(\mathbf{x}, t, \boldsymbol{\mu}) := \sum_{i=1}^{N_s N_t} \sum_{j=1}^{N_\mu} S_{ij} \phi^i(\mathbf{x}, t) \psi^i(\boldsymbol{\mu}) \,, \tag{16}$$

in which the set of functions $\phi_i(\mathbf{x}, t)$ forms a representation basis in the space-time domain, $(\mathbf{x}, t) \in \Omega \times [0, T]$. This allows applying exactly what has been described

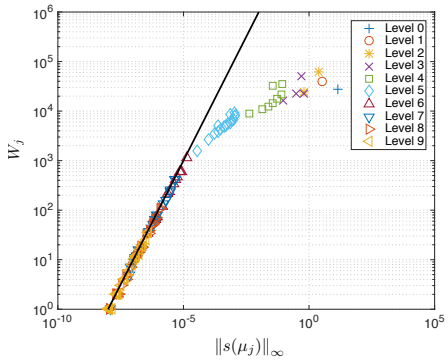


Fig. 4: The surplus norm of surplus functions $s(\boldsymbol{x}, \mu_j)$ correlates linearly with the amount of computational work W_j required to compute direct solutions M_{ij} using the predictions $\bar{M}_{ij}$ as initial guess in the nonlinear solver. The computational work is measured by the number of nonlinear iterations required by the direct solver to compute a solution. Since the computational work decays linearly with the magnitude of the surplus functions, and the latter decays exponentially with the order of the spectral approximation, we observe that the overall computational work per hierarchical level decreases when the hierarchical sampling reaches its asymptotic convergence regime.

in section 2. In particular, a subspace for the hierarchical surpluses may be found as shown in section 2.3.

An alternative approach consists in splitting space and time into separated dimensions, which yields:

$$u^h(\mathbf{x}, t, \boldsymbol{\mu}) := \sum_{i=1}^{N_s} \sum_{j=1}^{N_t} \sum_{k=1}^{N_\mu} S_{ijk} \phi^i(\mathbf{x}) \varphi^j(t) \psi^k(\boldsymbol{\mu}) \,, \tag{17}$$

where the set of functions $\phi^i(\mathbf{x})$, $\varphi^j(t)$ and $\psi^k(\boldsymbol{\mu})$ form a representation basis in the physical space $\mathbf{x} \in \Omega$, time interval $t \in [0, T]$ and parametric space $\boldsymbol{\mu} \in \mathcal{M}$, respectively. The scalars S_{ijk} are representation coefficients forming a third-order tensor.

Both Eq. (16) and (17) are in principle equivalent, as they yield a total complexity of $N_s \times N_t \times N_\mu$. However, Eq. (17) allows seeking for a tensor subspace, which is likely to be more compact in terms of representation than its matrix counterpart. We shall elaborate on the tensor subspace approach in next lines. For the sake of clarity, we shall denote a tensor $\mathbb{A}$ with values in the field $\mathbb{K}^{n_1 \times n_2 \times n_3}$ either by $\mathbb{A}_{|n_1,n_2,n_3}$ (the first time it is introduced) or simply $\mathbb{A}$, in subsequent appearances.

Suppose that the hierarchical collocation approach allowing for a sampling of the parameter domain is applied exactly in the same manner as described in section 2. As a result, we get a three-dimensional tensor $\mathbb{S}_{|n_1,n_2,n_3}$, containing the hierarchical surpluses, where n_1 is typically the number of degrees of freedom related to the space

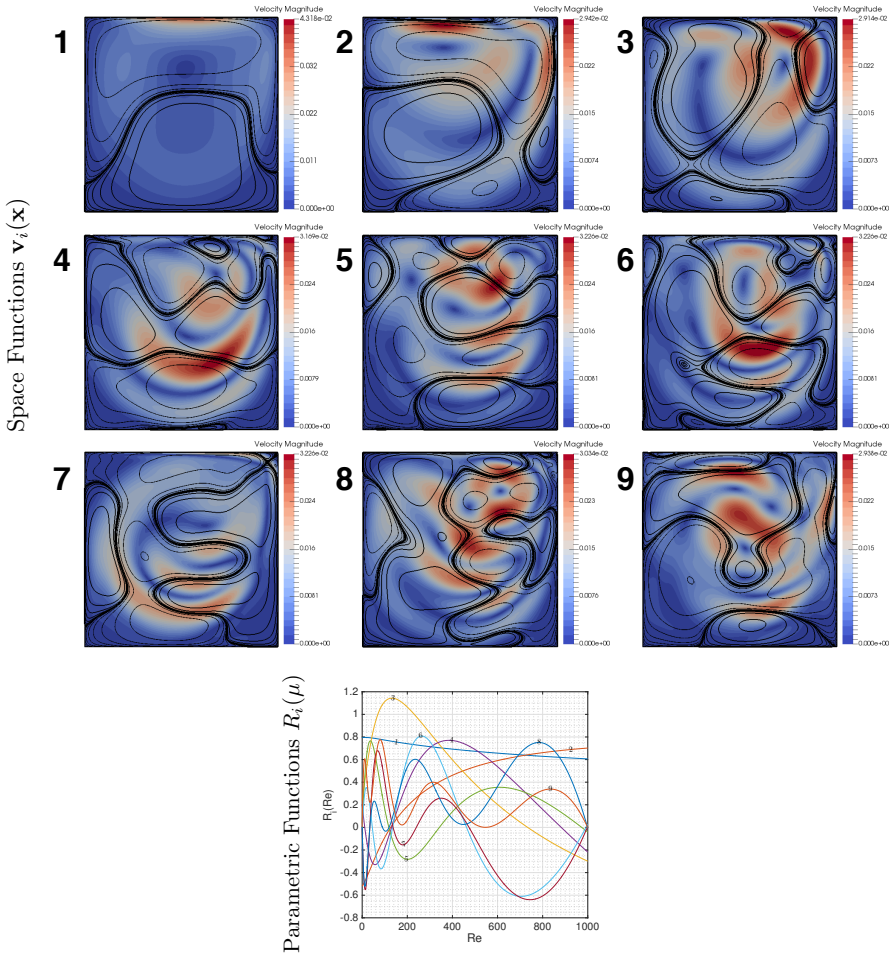


Fig. 5: Sparse Subspace Learning solution for the parametric lid-driven cavity steady state laminar flow for Reynolds number between 0 and 1000. The space modes are given by $\mathbf{v}_i(\mathbf{x}) = \sum_n \mathbf{U}_{ni}\phi_n(\mathbf{x})$. The first nine modes are visualized by streamlines and intensity of the corresponding field. The corresponding parametric functions are also reported. These are defined as $R_i(\mu) = \sum_n V_{mi}\psi_m(\mu)$. In this example the tolerance criterion used for the direct solver convergence was 10^{-8}, while the truncation criterion adopted for the determination of the numerical rank r in the irsvd algorithm is 10^{-7}.

discretization, n_2 is the number of time steps and n_3 is the total number of collocation points at convergence. We seek a tensor decomposition of rank-(r_1, r_2, r_3) given by

$$\mathbf{S} = \sum_{i=1}^{r_1}\sum_{j=1}^{r_2}\sum_{k=1}^{r_3}\sigma_{ijk}\mathbf{u}_1^i \otimes \mathbf{u}_2^j \otimes \mathbf{u}_3^k\,, \tag{18}$$

which using the tensor multiplication [[**?**], sec. 2.5], can also be written as follows:

$$\mathbb{S} = \mathbb{C} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \,, \tag{19}$$

where the factor matrices $\mathbf{U}_{1|n_1 \times r_1}$, $\mathbf{U}_{2|n_2 \times r_2}$ and $\mathbf{U}_{3|n_3 \times r_3}$ are usually orthonormal and define linear transformations in each direction. They can be regarded as the singular vectors in each direction. On the other hand, $\mathbb{C}_{|r_1 \times r_2 \times r_3}$ is called the core tensor, and its entries express the level of interaction between the different singular vectors.

A tensor decomposition such as Eq. (19) is usually computed with the `hosvd` algorithm [**?**], based on the application of the `svd` on successive matricizations of the original data. Briefly:

$$\mathbf{S}^{(d)} = \mathbf{U}_d \mathbf{\Sigma}_d \mathbf{V}_d^*, \quad d = 1, 2, 3. \tag{20}$$

The core tensor is computed:

$$\mathbf{\Sigma}^{(d)} = \mathbf{\Sigma}_d \mathbf{V}_d^* (\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_{d-1} \otimes \mathbf{U}_{d+1} \otimes \cdots \otimes \mathbf{U}_D). \tag{21}$$

Which in our case means that the core can be computed either as

$$\mathbf{\Sigma}^{(1)} = \mathbf{\Sigma}_1 \mathbf{V}_1^* (\mathbf{U}_2 \otimes \mathbf{U}_3), \tag{22}$$

$$\mathbf{\Sigma}^{(2)} = \mathbf{\Sigma}_2 \mathbf{V}_2^* (\mathbf{U}_1 \otimes \mathbf{U}_3), \quad \text{or} \tag{23}$$

$$\mathbf{\Sigma}^{(3)} = \mathbf{\Sigma}_3 \mathbf{V}_3^* (\mathbf{U}_1 \otimes \mathbf{U}_2). \tag{24}$$

Applying the `hosvd` may be expensive in terms of both memory requirements and computational cost, especially if the input data does not fit in fast memory. For this reason, we propose to replace the standard `svd` by the the `irsvd`, already presented in section A.2. In this manner, the factor matrices can be updated in order to account for the data stream.

3.1 Application to Sheet Metal Forming Simulation

In this example we apply SSL to transient stamping simulation of a hotformed automotive b-pillar. The objective is to assess the process sensitivity to the friction between the metal blank and the stamping tooling (punch and dye). Then numerical solution is sought in $\Omega \times \mathcal{T} \times \mathcal{M}$, where the space domain Ω is shown in figure 6, the time interval is defined as $\mathcal{T} \equiv [0, 8][s]$ and $\mathcal{M} \equiv [0.05, 0.3]$ is the parametric space of friction coefficients. Frictional contacts are modeled using Coulomb's friction law. Direct simulations were run on 4 hierarchical levels using the commercial software Pam-Stamp 2G. The metal blank is modeled using quadrilateral shell elements with 1.5mm initial uniform thickness. The stress-strain relation for steel is modeled by means of Krupkowsky's law taking into account the strain hardening during the metal forming process. The punch and the dye are assumed as rigid elements. From each simulation we export displacements, plastic strain, thickness and thinning fields, each requiring a separate tensor approximation.

For the displacement field, for instance, the final solution numerical rank is $(6, 5, 3)$, ensuring reasonable accuracy for this application (less than 1% error on the predicted displacement). Other fields produce similar results. Figure 6 shows space, time and parametric modes for the displacement field, while three particular solutions for the thinning are shown in figure 7.

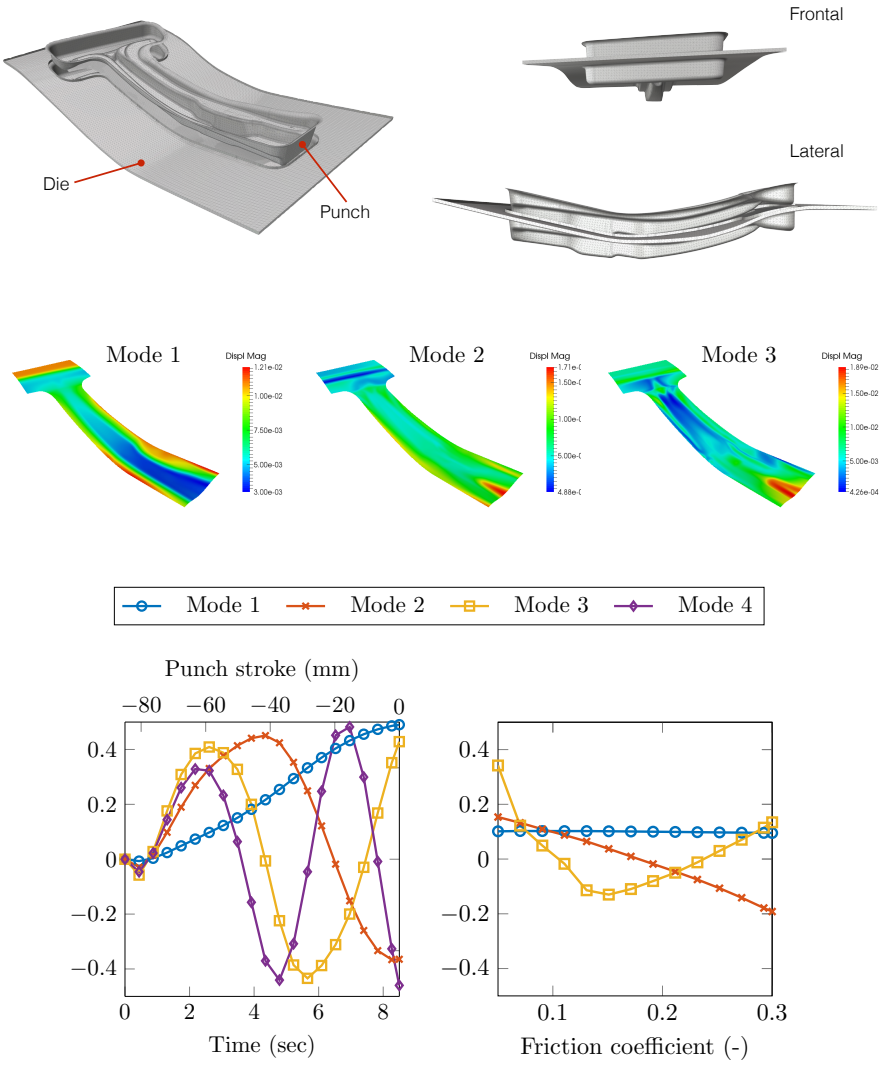


Fig. 6: Order reduction of the stamping solution: normalized space, time and parameter modes of the displacement field.

4 Extension to high-dimensional parametric problems

In higher dimensional problems the extension of hierarchical sampling is non trivial. Sampling high-dimensional spaces using the strategy described so far is limited by the curse of dimensionality. Indeed, the sampling points at a hierarchical level k would be simply given by the tensor product of point samplings $\mathcal{P}_d^{(k)}$ along each

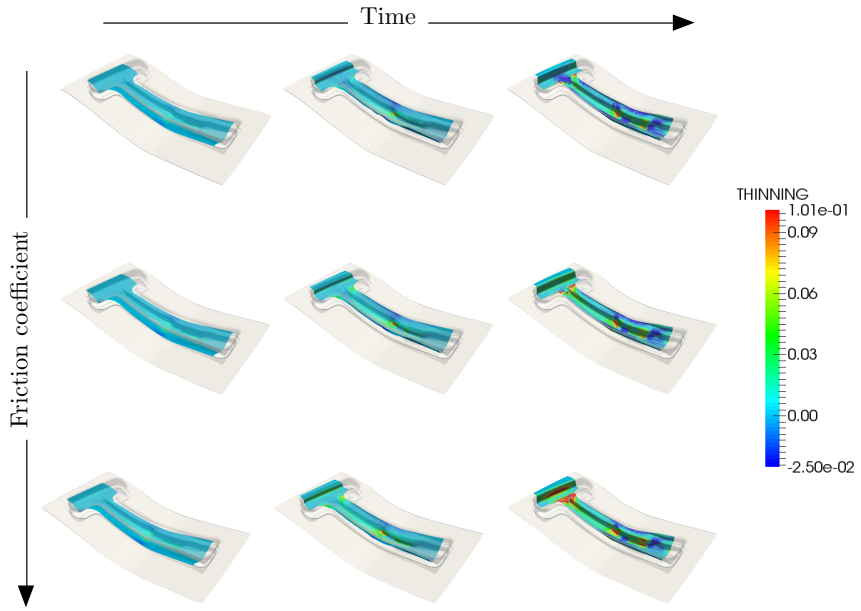


Fig. 7: The figures shows the solutions for three different friction coefficients 0.05 (top), 0.175 (middle) and 0.3 (bottom) at times $t = 2.32$ s (left), $t = 5.41$ s (center) and $t = 8.50$ s (right).

dimension d. Assuming D dimensions, this writes:

$$\mathcal{P}^{(k)} = \mathcal{P}_1^{(k)} \otimes \mathcal{P}_2^{(k)} \otimes \ldots \mathcal{P}_D^{(k)} \equiv \bigotimes_{d=1}^{D} \mathcal{P}_d^{(k)}$$

Given that each point set is the union of all previous hierarchical refinements

$$\mathcal{P}_d^{(k)} = \bigcup_{h=0}^{k} \mathcal{H}_d^{(h)},$$

The D-dimensional hierarchical refinement given by the tensor product rule would be :

$$\mathcal{H}^{(k)} \equiv \mathcal{P}^{(k)} \setminus \mathcal{P}^{(k-1)} = \bigcup_{h=1}^{D} \left[\mathcal{H}_h^{(k)} \otimes \bigotimes_{\substack{d=1 \\ d \neq l}}^{D} \mathcal{P}_d^{(k-1)} \right] \cup \bigotimes_{d=1}^{D} \mathcal{H}_d^{(k)} \quad \forall k \in \mathbb{N}^+$$

The tensor product rule implies that hierarchical refinements grow exponentially with the dimension D of the problem. To alleviate this issue we adopt Smolyak's

rule to construct higher dimensional refinement sets [**?**]. This replaces the traditional tensor product rule by the following expression:

$$\mathcal{H}^{(k)} \supset \mathcal{H}_s^{(k)} = \bigcup_{|\mathbf{h}|_1=k} \bigotimes_{d=1}^{D} \mathcal{H}_d^{(h_d)}, \forall k \in \mathbb{N}^+ \tag{25}$$

with $\mathbf{h} \equiv \{h_1, \ldots, h_d, \ldots, h_D\}$. The complexity Smolyak's rule grows polynomially with the dimensionality of the parametric hypercube, instead of exponentially, while retaining the same accuracy on the approximation of the function, provided that the high order mixed derivatives are bounded [**?**]. Therefore the method is not subject to the curse of dimensionality at least in the space of smooth functions. This concept has been successfully used in the framework of sparse grid interpolation and quadrature [**?**], with a variety of different basis functions including hierarchical polynomials and wavelet functions [**?**] and extended to dimensionally adaptive strategies [**?**]. Note that changing the ℓ_1 norm for a different norm in equation (25) yields different sampling strategies. Using the infinity norm (maximum norm) the classical tensor-product sampling is obtained while the Euclidean norm gives rise to the hyperbolic-cross sampling [**?**].

4.1 Application to Stokes flow around parametric NACA four digits airfoil

In this section we show an application of the Sparse Subspace Learning method to the problem of steady state viscous incompressible and inertialess flow around a NACA four-digits airfoil. The geometry of the airfoil is known in analytical closed form as a function of four parameters: the chord c, the maximum thickness t, the maximum camber m and the position of maximum camber as a fraction of the chord p.

In dimensionless form, the problem only depend on three parameters if the chord is chosen as reference unit length. Therefore the parametric solution of this problem is sought in the three-dimensional space

$$\{t, m, p\} \in [0.06, 0.15] \times [0, 0.1] \times [0.34, 0.5]$$

The velocity and pressure fields are governed by Stokes equations:

$$\begin{cases} \Delta \mathbf{v} - \nabla p &= 0 \\ \nabla \cdot \mathbf{v} &= 0 \end{cases} \tag{26}$$

paired with appropriate boundary condition prescribing no-slip at the airfoil boundary $\mathbf{v}(\mathbf{x} = \mathbf{x}_b) = \mathbf{0}$ and uniform asymptotic velocity $\mathbf{v}_\infty = cos(\alpha)\mathbf{i} + sin(\alpha)\mathbf{j}$; with the angle of attack $\alpha = 10^\circ$ for this case. The airfoil shape variation is taken into account through the mapping

$$\mathbf{x}(\mathbf{x}_0) = \mathbf{x}_0 + \mathbf{u}(\mathbf{x}_0)$$

where $\mathbf{x}_0$ is the coordinate in a reference (undeformed) domain, $\mathbf{x}$ is the physical coordinate in the transformed domain and $\mathbf{u}$ is a displacement field. In practice the mapping is found through the solution of the elliptic problem for the $\mathbf{u}$:

$$c_1\mathbf{u} + c_2\Delta_{\mathbf{x}_0}\mathbf{u} + c_3\nabla_{\mathbf{x}_0}[\nabla_{\mathbf{x}_0} \cdot \mathbf{u}] = \mathbf{0} \tag{27}$$

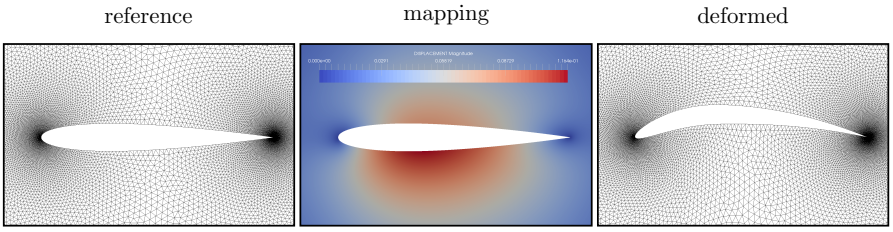


Fig. 8: Airfoil shape variation with respect to a reference shape (NACA 0012 airfoil) are take into account by deforming a reference triangular mesh (leftmost panel). The nodes in the original mesh are moved according to a displacement field $\mathbf{u}$ solution of the elliptic problem (27) with imposed Dirichlet boundary condition prescribing the right shape on the airfoil boundary (central panel). An exemple of a typical mesh is shown in the rightmost panel.

with prescribed Dirichlet boundary conditions on the airfoil boundaries in order impose the correct shape to the deformed domain. The subscript $\mathbf{x}_0$ indicates that the derivatives are taken with respect to the coordinates in the reference domain.

The reference shape is chosen as the symmetric profile NACA 0012. The reference domain is then meshed using triangular elements. Equation (27) is solved on this domain to find the mapping $\mathbf{u}$ corresponding to a given shape of the family NACA 4-digits airfoils and the reference mesh is deformed accordingly, as shown in figure 8. The coefficients appearing in this equation are chosen on empirical basis as $c_1 = 50$, $c_2 = 1$, $c_3 = 24.5$. The values are selected in order to avoid severe distorsion of the mesh. The Stokes problem is then solved on the deformed mesh using Crouziex-Raviart mixed finite elements formulation.

There are two bottlenecks associated to this problem in the framework of projection based parametric reduced order modeling:

- The parametric mapping $\mathbf{u}(\mathbf{x}_0, t, m, p)$ needs to be determined from the solution of equation (27). The difficulty in this step is associated to the parametrization of the Dirichlet boundary conditions describing the airfoil shape that are not expressed in a tensor format and an approximate decomposition must be sought.
- Even if a separated variable approximation of $\mathbf{u}(\mathbf{x}_0, t, m, p)$ can be obtained with reasonable accuracy, the weak form associated to problem (26) is not necessarily affine with respect to parameters $\{t, m, p\}$.

A strategy to recover affine approximations in terms of geometrical parameters can be found in [**?**].

In this case the use of collocation does not require an affine approximation of the linear operators arising from the problem discretization, since the proposed method only relies on the output of the deterministic solver. This takes values of $\{t, m, p\}$ as input and computes the Stokes flow around the corresponding airfoil using a triangular mesh generated through equation (27). A sparse and low rank approximation of both the flow solution and the mapping are constructed simultaneously.

The sampling algorithm is run until all hierarchical surpluses fall below 10^{-8} and the truncation threshold for the rank was taken as 10^{-15}. In figure 9 we present

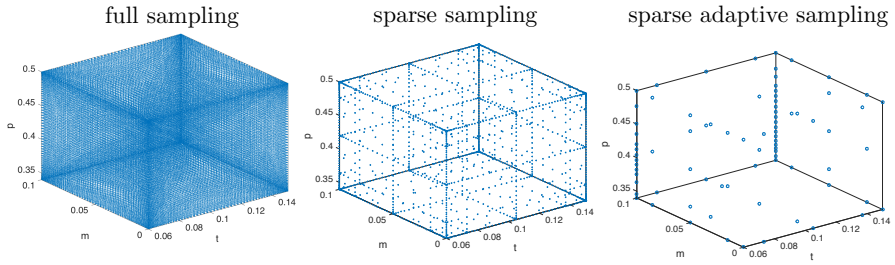


Fig. 9: Comparison between tensor product sampling strategy (leftmost panel), sparse grid sampling using Smolyak's rule (central panel) and sparse adaptive sampling (rightmost panel). Note how the sampling along p is automatically refined in proximity of the values $m = 0.1$. This is because for $m = 0$ (symmetric profile) the position of maximum camber p does not affect the shape profile. On the contrary for cambered airfoils the value of the parameter p has a strong effect on the shape and therefore on the flow solution, therefore more sampling points are needed in this region.

a comparison between three possible sampling strategies. With respect to classical tensor product of 1D sampling points, Smolyak's rule requires only a 0.6% of all points and adaptive Smolyak's rule only 0.03%. The final solution approximate rank (for the prescribed precision) is 16. To assess the validity of the solution we compared results of the reduced parametric model to the results of the direct solver for some parameter combinations giving specific airfoils in the NACA 4-digits family. Results are presented in figure 10 and confirm that the accuracy of the model is around 10^{-8}. Note that error is sensibly smaller close to the location of the sampling points.

4.2 Application to crash simulations

Parametric modeling is of capital importance in simulation based engineering applications such as process and product optimization, identification, control and uncertainty quantification. Reduced order modeling offers a practical way to alleviate the curse of dimensionality that is inevitably encountered in high-dimensional parametric problems. In industrial practice the need of simulation code certification has somehow slowed down the process of integrating high fidelity simulation software with reduced order modeling tools. This is partly due to the intrusiveness of most of reduced order modeling approaches, which inevitably require modification of the original code to a certain extent.

In this last section we show the results of parametric crash simulations performed using the commercial code Pam Crash. The reason behind the choice of this problem is very well related with the intrusiveness issue. In crash simulation, parametric modeling is extremely important in order to have a quick assessment of parameter sensitivity and performance/safety estimators before the actual physical model is built and tested. The "digital twin" approach is a cost-wise efficient way to have a quick

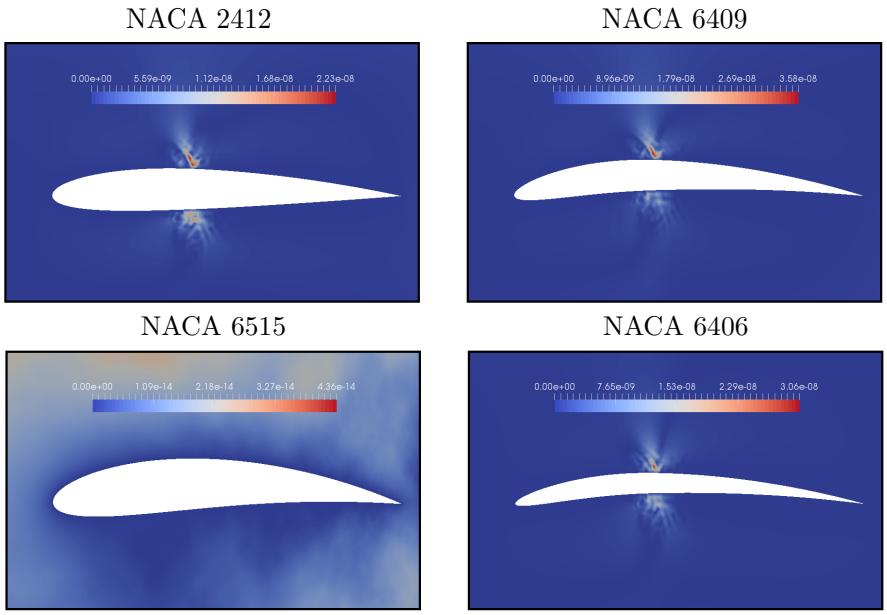


Fig. 10: Error maps of the reduced order parametric solution with respect to direct solutions for specific airfoils shapes of the NACA four-digits family. The error is globally around 10^{-8} everywhere in the parametric space except in proximity of the collocation points where the error is remarkably smaller.

sweep of the parameter state space and identify the optimal configurations in the pre-design step. These will be eventually verified using high performance simulation and ultimately certified by experimental testing.

Crash simulation code include binary actions that are external to the core PDE solver, such as plasticity or contact detection. In particular the last is extremely time consuming and can take up to 90% of the simulation time. In general such "control actions" are present in most simulation software, which is seldom limited to a simple core PDE solver.

Traditional reduced order modeling requires a redefinition of the control actions. For instance the very concept of contact detection changes when the problem is formulated in terms of a reduced basis with global support, due to the intrinsically local nature of contacts.

A non-intrusive approach allows to leave control actions to a lower level, inside the deterministic solver, since the reduced parametric model is built only on the solver output. For the same reason, it is also possible to build reduced models for specific quantity of interests extracted from simulation results.

In the example reported in this section, crash simulations were run using the commercial software Pam Crash. The tested configuration correspond to the NCAP Offset Deformable barrier frontal impact, shown in figure 11. The car is driven at 64km/h

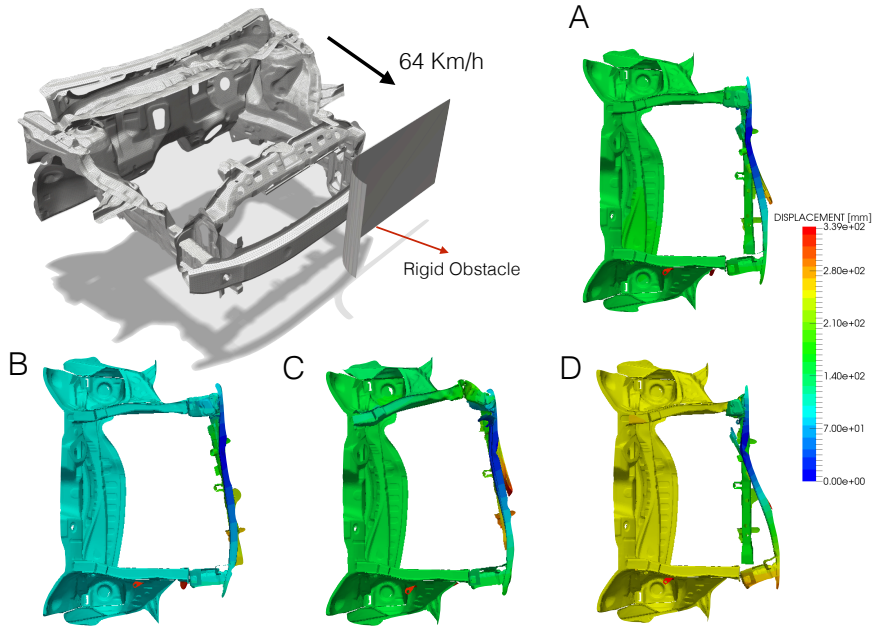


Fig. 11: Parametric simulation of the crash model for the NCAP Offset Deformable barrier frontal impact test. Final deformation corresponding to different combination of the parameters : (A) $\mu_1 = 2.34\,\text{mm}$, $\mu_2 = 2.09\,\text{mm}$, $\mu_3 = 2.362.34\,\text{mm}$, $\mu_4 = 1.61\,\text{mm}$; (B) $\mu_1 = 3.74\,\text{mm}$, $\mu_2 = 2.09\,\text{mm}$, $\mu_3 = 3.78\,\text{mm}$, $\mu_4 = 1.61\,\text{mm}$; (C) $\mu_1 = 3.74\,\text{mm}$, $\mu_2 = 0.84\,\text{mm}$, $\mu_3 = 3.78\,\text{mm}$, $\mu_4 = 0.64\,\text{mm}$; (D) $\mu_1 = 2.34\,\text{mm}$, $\mu_2 = 0.84\,\text{mm}$, $\mu_3 = 2.34\,\text{mm}$, $\mu_4 = 0.64\,\text{mm}$.

and with 40% overlap into a deformable barrier which represents the oncoming vehicle. The test replicates a crash between two cars of the same weight, both travelling at a speed of 50km/h. In particular safety can be assessed with respect to the geometrical parameters in the model. In this case we considered four parameters, $\boldsymbol{\mu} \equiv \{\mu_1, \mu_2, \mu_3, \mu_4\}$, corresponding to the thicknesses of different parts (inner rail, outer rail, lower-outer rail and front frame).

The reduced parametric displacement field is sought in the tensor form

$$\mathbf{u}^h(\mathbf{x}, t, \boldsymbol{\mu}) = \sum_{i=1}^{r_s} \sum_{j=1}^{r_t} \sum_{k=1}^{r_\mu} C_{ijk} \phi^i(\mathbf{x}) \varphi^j(t) \psi^k(\boldsymbol{\mu}), \tag{28}$$

Simulation were scripted using Python and a Matlab generic interface was used to recover the simulation results and build the reduced model. Globally, 401 simulations were run in parallel, each taking approximately one hour on 32 cores. The final deformations corresponding to different choices of the parameters are shown in figure 11.

The model was exported using the *pxdmf* format for canonical tensor representations developed in the framework of PGD [**?**]. This allows the visualization and the post-processing of the solution in real time using open source software Paraview. This solution can be easily explored for sensitivity analysis or uncertainty quantification with respect to the parameters.

5 Concluding remarks

We showed the application of hierarchical adaptive sampling in the parametric space combined with an incremental tensor approximation technique in order to learn the low-rank and sparsity features characterizing the solution of parametric models. We tested the proposed approach through numerical experimentation on different models, including time dependent and multi-parametric problems. Results show that the collocation strategy can be easily integrated with existing deterministic solvers in a non-intrusive way, as the method does not require access and manipulation of the solver internal operators or routines. This feature enables reduced order modeling for problems which are not straightforwardly compatible with traditional projection-based approaches requiring the appropriate format in the problem setting. Approximating a parametric solution in explicit form not only requires the construction of a reduced basis but also the determination of the parametric modes. When these have a sparse representation, the number of coefficients needed to accurately describe the functional dependency of the solution in the parametric space is typically small, although this is in general higher than the numerical rank of the approximation. Therefore, in agreement with other studies [**?**], we observed that if the number of direct runs sampling is fixed, the offline/online approach yields a more accurate solution (when the structure of the problem makes it possible) than an explicit offline solution. However, as pointed out in [**?**], in many applications the hierarchical collocation approach is sufficient to have reasonable accuracy for an explicit representation of the solution, or a given quantity of interest, without the need of solving a reduced system each time a new query is demanded.

In many engineering applications, the proposed approach is particularly suitable for delivering fast performance estimators, allowing for quick parametric sweeps so as to assess sensitivity with respect to the parameters, find optimality or evaluate and quantify uncertainty in the data.

A Incremental Random Singular Value Decomposition

A.1 Randomized singular value decomposition

Suppose we are given the input data matrix $\mathbf{S}$, which in our case contains the hierarchical surpluses, and assume that a rank-r approximation like in Eq. (7) wants to be computed. There are two main ideas behind the `rsvd`:

- The first is to realize that a partial SVD can be readily computed provided that we are able to construct a low-dimensional subspace that captures most of the range of the input data matrix.
- The second is to observe that such low-dimensional subspace can be computed very efficiently using a random sensing method.

Let us start with the first of the aforementioned ideas. Suppose that we are given a matrix $\mathbf{Q}_{|m\times p}$ with $r \leq p \ll n$ orthonormal columns such that the range of $\mathbf{S}$ is well captured, i.e.:

$$\|\mathbf{S} - \mathbf{Q}\mathbf{Q}^*\mathbf{S}\| \leq \varepsilon, \tag{29}$$

for some arbitrarily small given tolerance ε. Then, the input data is restricted to the subspace generated by its columns, that is: $\mathbf{B}_{|p\times n} = \mathbf{Q}^*\mathbf{S}$. Observe at this point that we implicitly have the factorization $\mathbf{A} \approx \mathbf{Q}\mathbf{B}$. Next, we compute an SVD factorization of the matrix $\mathbf{B} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$, where factors are defined as $\tilde{\mathbf{U}}_{|p\times p}$, $\tilde{\mathbf{\Sigma}}_{|p\times p}$ and $\tilde{\mathbf{V}}_{|n\times p}$. This operation is much less expensive than performing the SVD on the initial data set because the rank is now restricted to the rows of $\mathbf{B}$. Finally, in order to recover the r dominant components, we define an extractor matrix $\mathbf{P}_{|p\times r}$ and set: $\mathbf{U} = \mathbf{Q}\tilde{\mathbf{U}}\mathbf{P}$, $\mathbf{\Sigma} = \mathbf{P}'\tilde{\mathbf{\Sigma}}\mathbf{P}$ and $\mathbf{V} = \tilde{\mathbf{V}}\mathbf{P}$. In summary, given $\mathbf{Q}$, it is straightforward to compute a SVD decomposition at a relatively low cost $O(mnp + (m+n)p^2)$.

Now we address the second question, that is, how to compute the matrix $\mathbf{Q}$. We first draw a random Gaussian test matrix, $\mathbf{\Omega}_{|n\times p}$. Then, we generate samples from the data matrix, i.e. $\mathbf{Y}_{|m\times p} = \mathbf{A}\mathbf{\Omega}$. Observe that if the rank of input data matrix was exactly r, the columns of $\mathbf{Y}$ would form a linearly independent set spanning exactly the range of $\mathbf{S}$, provided that we set $p = r$. Since in general the *true* rank will be greater than r, we must consider an *oversampling* parameter by setting $p = r + \alpha$. This will produce a matrix $\mathbf{Y}$ whose range has a much *better chance* of approximating well the range of the input data matrix. Finally, $\mathbf{Q}$ can be obtained from the orthogonalization of $\mathbf{Y}$. In fact, it can be shown that the following error bound is satisfied

$$\|\mathbf{S} - \mathbf{Q}\mathbf{Q}^*\mathbf{S}\| \leq \left[1 + 9\sqrt{p\min\{m,n\}}\right]\sigma_{r+1}, \tag{30}$$

with a probability in the order of $O(1 - \alpha^{-\alpha})$. That is, the failure probability decreases superexponentially with the oversampling parameter [**?**].

Remark 1 (On the optimal decomposition) Observe that the standard SVD produces $\mathbf{Q}_{|m\times r}$ such that

$$\|\mathbf{S} - \mathbf{Q}\mathbf{Q}^*\mathbf{S}\| = \sigma_{r+1},$$

but at a higher cost $O(mn\min\{m,n\})$.

A prototype version of the randomized SVD is given in the Algorithm 4.

Algorithm 4 Randomized singular value decomposition: `rsvd`

Require: $\mathbf{S}_{|m\times n}$, r, α ▷ Data matrix, rank and oversampling parameter
Set $p = r + \alpha$
1: Draw $\mathbf{\Omega}_{|n\times p}$ ▷ Random Gaussian test matrix
2: Generate samples $\mathbf{Y}_{|m\times p} = \mathbf{S}\mathbf{\Omega}$
3: Compute $\mathbf{Q}_{|m\times p} = \texttt{orthogonalize}(\mathbf{Y})$ ▷ Captures range of $\mathbf{S}$
4: Restrict $\mathbf{B}_{|p\times n} = \mathbf{Q}^*\mathbf{S}$
5: Compute $[\tilde{\mathbf{U}}_{|p\times p}, \mathbf{\Sigma}_{|p\times p}, \mathbf{V}_{|n\times p}] = \texttt{svd}(\mathbf{B})$ ▷ Standard deterministic SVD
6: Set $\mathbf{U}_{|m\times p} = \mathbf{Q}\tilde{\mathbf{U}}$
return $\mathbf{U}_{|m\times r}, \mathbf{\Sigma}_{|r\times r}, \mathbf{V}_{|n\times r}$ ▷ Retain r first components

Neglecting the cost of generating the Gaussian random matrix $\mathbf{\Omega}$, the cost of generating the matrix $\mathbf{Q}$ is in the order of $O(mnp + mp^2)$ flops. In consequence, the computational cost of the entire `rsvd` procedure remains as $O(mnp + (m+n)p^2)$. The algorithmic performance of the `rsvd` can be further improved by introducing a number of refinements at the price of worsening slightly the error bounds. In particular, the most expensive steps in the `rsvd` algorithm consist in forming matrices $\mathbf{Y}$ and $\mathbf{B}$, which require in the order of $O(mnp)$ flops. The first can be reduced to $O(mn\log(p))$ by giving some structure to the random matrix $\mathbf{\Omega}$, while the second can be reduced to $O((m+n)p^2)$ via row extraction techniques, which leaves the total cost $O(mn\log(p) + (m+n)p^2)$. The interested reader can find further details on these refinements as well as on their impact on the assessment in [**?**].

A.2 Incremental randomized singular value decomposition

In this section we present an incremental variant of the randomized SVD algorithm, discussed in section A.1. The objective is twofold: (i) to be able to learn a subspace for the hierarchical surpluses as they are streamed from the sparse sampling procedure; (ii) to perform it at a computational cost that scales reasonably with the number of samples.

Let us assume that we want to compute a rank-r approximation of some streamed data, and that we have chosen an oversampling parameter α such that $p = r+\alpha$, as in section A.1. Let us denote by $\mathbf{S}_{0|m\times n}$ the *old* data matrix, whereas $\mathbf{S}_{|m\times n'}$ is the *new* data columns such that the total data is now $\mathbf{S}_{1|m\times(n+n')} = [\mathbf{S}_0|\mathbf{S}]$. We would like to compute an approximated SVD decomposition $\mathbf{S}_1 \approx \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^*$ at a cost which is roughly independent on n, the number of columns of the old data. For the sake of completeness, recall that $\mathbf{U}_{1|m\times p}$, $\mathbf{\Sigma}_{1|p\times p}$ and $\mathbf{V}_{1|(n+n')\times p}$.

In order to do so, suppose that we are given a non-truncated SVD approximation of the old data, i.e. $\mathbf{S}_0 \approx \mathbf{U}_0\mathbf{\Sigma}_0\mathbf{V}_0^*$, with $\mathbf{U}_{0|m\times p}$, $\mathbf{\Sigma}_{0|p\times p}$ and $\mathbf{V}_{0|n\times p}$. Suppose that we also dispose of the matrix of random samples $\mathbf{Y}_{0|m\times p}$. Then, in order to account for the new data we only need to generate a random Gaussian test matrix $\mathbf{\Omega}_{|n'\times p}$ and perform a small product which only involves the new data:

$$\mathbf{Y}_1 = \mathbf{Y}_0 + \mathbf{S}\mathbf{\Omega}. \tag{31}$$

The matrix $\mathbf{Q}_{1|m\times p}$ can be obtained from the orthogonalization of $\mathbf{Y}_1$ at a cost that remains stable, as it does not depend on n nor n'. Next, input data has to be restricted to the range of $\mathbf{Q}_1$. Recalling that we already dispose of a non-truncated SVD approximation of the old data:

$$\mathbf{B}_1 \approx \mathbf{Q}_1^*\left[\,\mathbf{U}_0\mathbf{\Sigma}_0\mathbf{V}_0^*\middle|\mathbf{S}\,\right] = \underbrace{\left[\,\mathbf{Q}_1^*\mathbf{U}_0\mathbf{\Sigma}_0\middle|\mathbf{Q}_1^*\mathbf{S}\,\right]}_{\tilde{\mathbf{B}}}\left[\begin{array}{c|c}\mathbf{V}_0^* & \mathbf{0}\\ \hline \mathbf{0} & \mathbf{I}_{n'\times n'}\end{array}\right], \tag{32}$$

where $\mathbf{I}_{n'\times n'}$ is the identity matrix of size n'. Similarly to section A.1, observe that Eq. (32) yields a factorization $\mathbf{S}_1 \approx \mathbf{Q}_1\tilde{\mathbf{B}}$. Hence, if we compute a SVD decomposition of the factor $\tilde{\mathbf{B}}$,

$$\tilde{\mathbf{B}} = \tilde{\mathbf{U}}\mathbf{\Sigma}_1\tilde{\mathbf{V}}^*, \quad \text{with } \tilde{\mathbf{U}}_{|p\times p},\ \mathbf{\Sigma}_{1|p\times p} \text{ and } \tilde{\mathbf{V}}_{|(p+n')\times p}, \tag{33}$$

we can conclude the algorithm by setting:

$$\mathbf{U}_1 = \mathbf{Q}_1\tilde{\mathbf{U}} \quad \text{and} \quad \mathbf{V}_1 = \left[\begin{array}{c|c}\mathbf{V}_0 & \mathbf{0}\\ \hline \mathbf{0} & \mathbf{I}_{n'\times n'}\end{array}\right]\tilde{\mathbf{V}}. \tag{34}$$

A prototype version of the incremental randomized SVD is given in the Algorithm 5.

Algorithm 5 Incremental randomized singular value decomposition: `irsvd`

Require: $\mathbf{S}_{|m\times n'}$, $\mathbf{U}_{0|m\times p}$, $\mathbf{\Sigma}_{0|p\times p}$, $\mathbf{V}_{0|n\times p}$, $\mathbf{Y}_{0|m\times p}$ ▷ Streamed data, old factors and old samples

1: Draw $\mathbf{\Omega}_{|n'\times p}$
2: Correct samples $\mathbf{Y}_{1|m\times p} = \mathbf{Y}_0 + \mathbf{S}\mathbf{\Omega}$
3: Compute $\mathbf{Q}_{1|m\times p} = \texttt{orthogonalize}(\mathbf{Y}_1)$
4: Form $\tilde{\mathbf{B}}_{|p\times(p+n')} = \left[\,\mathbf{\Sigma}_0\middle|\mathbf{U}_0^*\mathbf{S}\,\right]$
5: Compute $[\tilde{\mathbf{U}}_{|p\times p}, \mathbf{\Sigma}_{1|p\times p}, \tilde{\mathbf{V}}_{|(p+n')\times p}] = \texttt{svd}(\tilde{\mathbf{B}})$
6: Set $\mathbf{U}_{1|m\times p}$ and $\mathbf{V}_{1|(n+n')\times p}$ as indicated in Eq. (4)

return $\mathbf{U}_1, \mathbf{\Sigma}_1, \mathbf{V}_1$ and $\mathbf{Y}_1$ ▷ Do not truncate: retain all p components

Observe that the cost of the `irsvd` algorithm is driven by $O((m+n)p^2)$ when choosing $n' \sim p$, while if one chooses $n' \gg p$, the cost matches the standard rSVD, that is $O(mn'p)$. A more detailed analysis of the flop count indicates that in fact, the only dependence on n of the algorithm is due to the cost of updating the right singular vectors in Eq. (34). On the other hand, the reader should keep in mind that, for the applications targeted in this paper, the number of rows of the input dataset (degrees of freedom after discretization of a PDE) is at least one or two orders of magnitude bigger than the number of columns (solution snapshots). As a consequence, the cost of the `irsvd` turns out to be roughly independent on n. A final consideration that should not be neglected is that, for data sets that do not fit in the core memory, the cost of transferring data from slow memory dominates the cost of the arithmetics. This can be generally avoided with the incremental algorithm presented in this section.

A.3 A numerical example: order reduction applied to the lid-driven cavity problem

In this section, we provide numerical evidence on the performance of the `irsvd`, as described in section A.2. In particular, we apply `irsvd` on the set of hierarchical surpluses, $\mathbf{S}_{\text{ldc}}$, coming from the solution of the lid-driven cavity problem, as described in 2.4. The size of the data matrix is $m = 14,082$ rows and $n = 513$ columns. An overkill value of the oversampling parameter is taken, $\alpha = r$ (i.e. $p = 2\,r$).

Firstly, we show that the low-rank singular value decomposition given by the `irsvd` tends to both the standard `svd` and the `rsvd` as the number of processed columns approaches the number of columns of the entire dataset. To that end, we choose a rank $r = 20$ and a fixed bandwidth $n' = 5$. Figure 14 shows the evolution of the singular values as the number of processed columns increases. It can be noticed that, in fact, after a relatively low number of columns are processed (say 20), the singular values are already very close to the reference ones. This is simply because when coupling `irsvd` with the hierarchical sampling the surpluses that come from higher hierarchical levels are naturally associated to the first singular vectors. On the contrary, lower hierarchical levels yield smaller surpluses, as the hierarchical sampling method converges. When the entire dataset is processed, the `irsvd` yields a SVD that matches the standard one, see Figure 12f.

In order to further assess the convergence of the `irsvd` towards the standard `svd` decomposition, the energy error between both decompositions is measured:

$$\varepsilon_r = \sqrt{\frac{\sum_{i=1}^{r} (\sigma_{\texttt{irsvd}} - \sigma_{\texttt{svd}})^2}{\sum_{i=1}^{r} \sigma_{\texttt{svd}}^2}}, \tag{35}$$

for a given rank r. Figure 13 shows the evolution of ε_r for several bandwidth. It can be observed that the bandwidth hardly influences the convergence results.

Next, the computational cost of the `irsvd` must be assessed. Figure 14a shows the runtime (denoted by τ) of the `irsvd`, i.e. Algorithm 5, as a function of the bandwidth. The runtime is computed as the average of three executions. Results confirm that, as discussed in section A.2, the computational cost is independent on the bandwidth size. Besides, it can be observed that greater ranks yield greater runtimes. In fact, the computational complexity should depend quadratically on the rank. This quadratic scaling is confirmed by Figure 14b, which shows the normalized rank $\tilde{r} = r/r_0$ (with $r_0 = 10$) against the normalized runtime $\tilde{\tau} = \tau/\tau_0$, where τ_0 is the runtime associated to r_0. It can be seen that for all bandwidth the normalized runtime scales super-linearly with the normalized rank (linear scaling is depicted for reference).

Finally, it is worth to highlight that in many practical applications the cost of `irsvd` turns out to be independent on n, the total number of columns of the data set. This is simply because usually $m \gg n$ and then the computational complexity reduces to $O(mp^2)$. In other words, the cost only starts being influenced by n when $n \sim m$. Figure 15 shows the runtime of each `irsvd` call, averaged over three runs. For the sake of clarity, runtimes have been normalized to their mean value, while the vertical axis scale is chosen so we can observe ±50% deviations from the mean. Results show that runtime deviates very few from the mean. Moreover, the cost of each call remains fairly constant as the number of processed columns increases, which confirms the discussion above.

Acknowledgments

The authors of the paper would like to acknowledge Jean-Louis Duval, Jean-Christophe Allain and Julien Charbonneaux from the ESI group for the support and data for crash and stamping simulations.

Compliance with Ethical Standards.

- The authors declare that they have no conflict of interest;
- The research does not involve neither human participants nor animals;
- All the authors are informed and provided their consent.

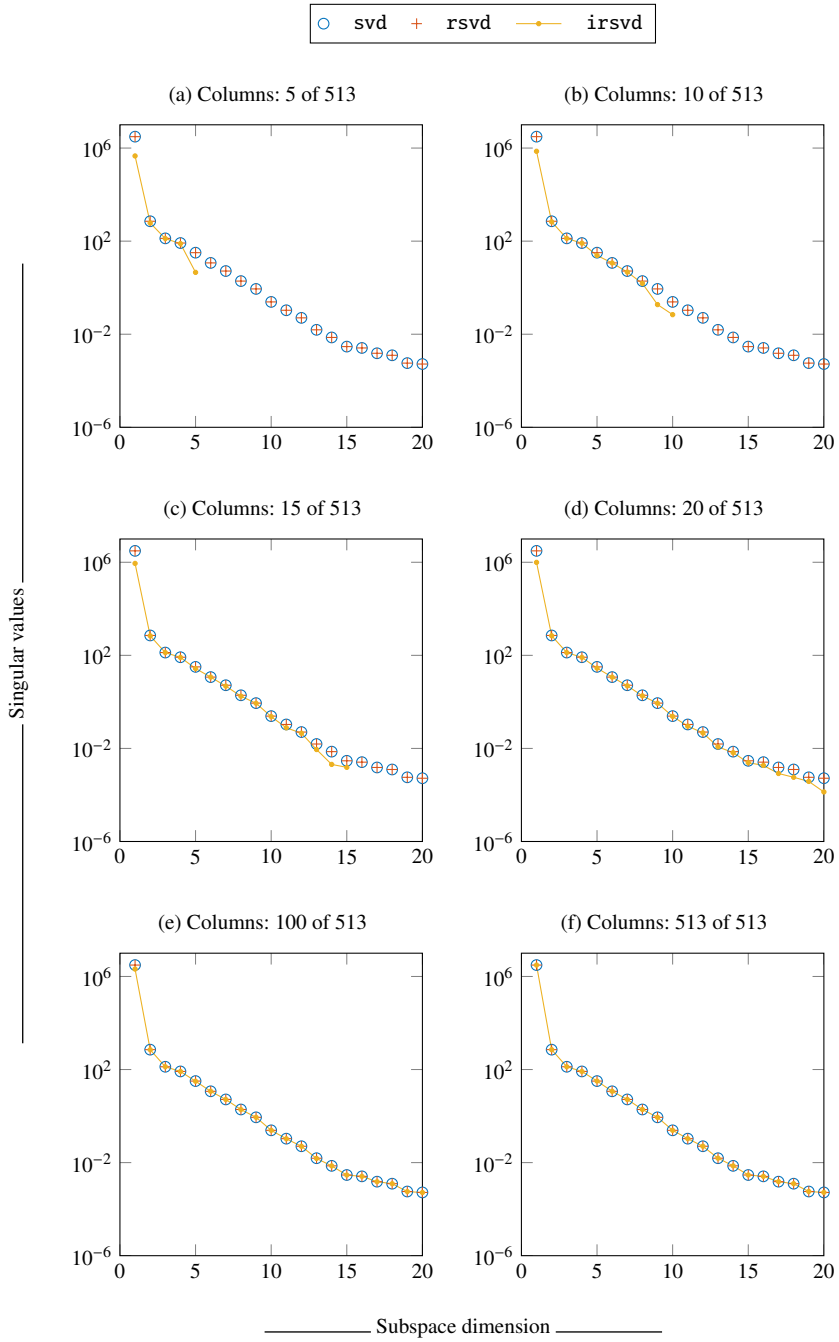


Fig. 12: Singular values evolution in terms of the cumulated number of columns processed by the `irsvd`. Comparison is made against reference results given by standard `svd` and `rsvd`, for rank $r = 20$ and bandwidth $n' = 5$.

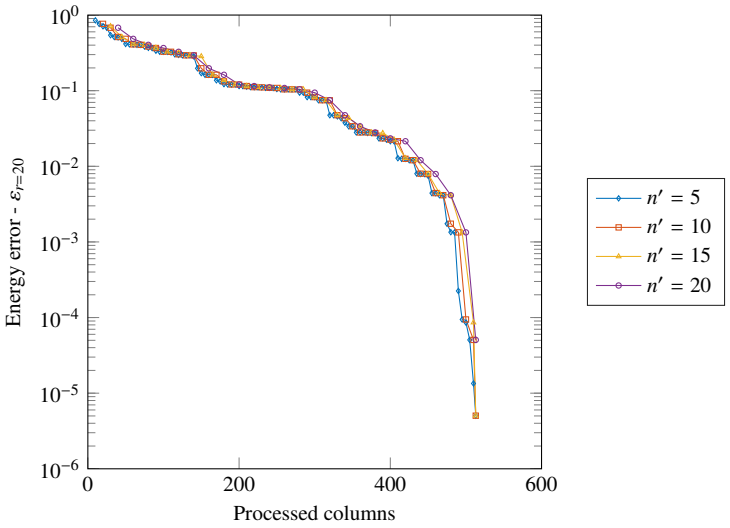
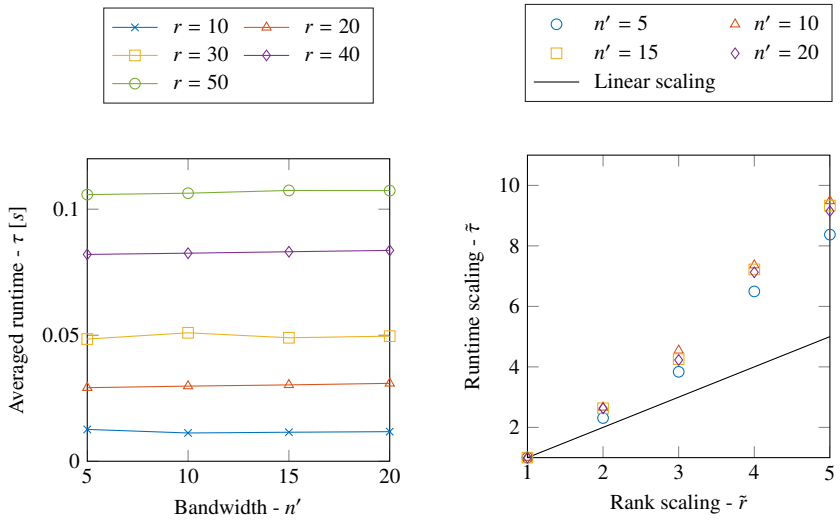


Fig. 13: Energy error measuring the convergence of the singular values, for fixed rank $r = 20$, as a function of the number of processed columns for different values of the bandwidth n'.



(a) Runtime is independent on the bandwidth for all rank

(b) Runtime scales super-linearly with the rank for all bandwidth

Fig. 14: Assessment of computational performances of `irsvd`: bandwidth independence and rank scaling.

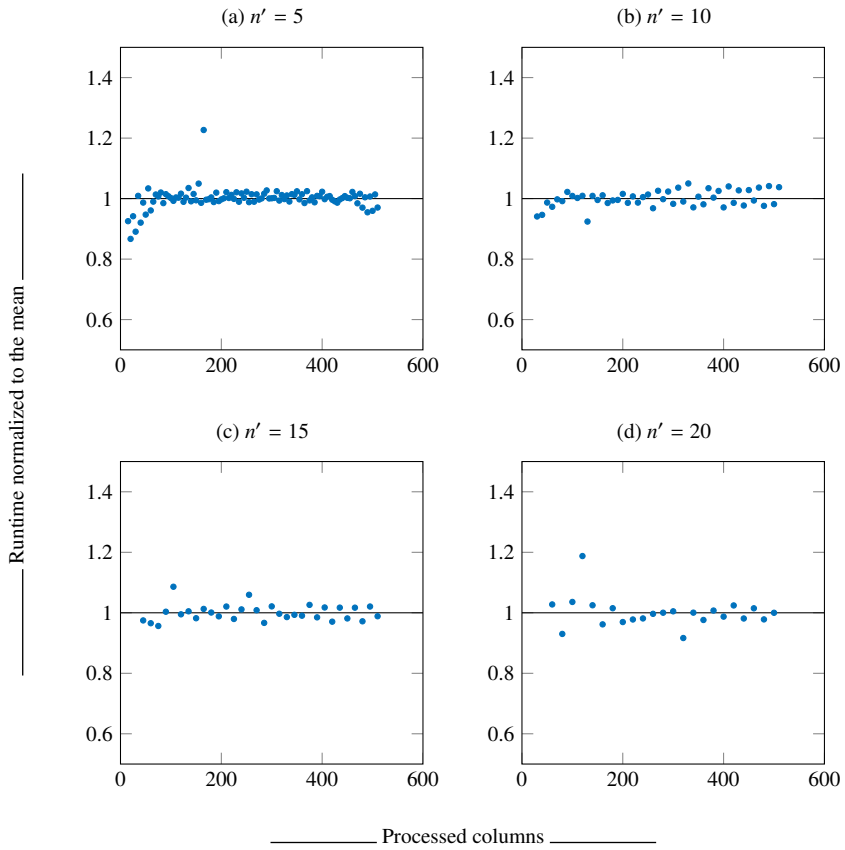


Fig. 15: Assessment of computational performances of `irsvd`: data size independence.